Requirements: Assignments have to be typeset in English. They have to be handed in by each student separately, except for exercises marked with (G), which should be solved and handed in by each (project) group. Please include your name on top of the first sheet.

How to describe algorithms: Whenever you are asked to describe an algorithm, you should present three things: the algorithm, a proof of its correctness, and a derivation of its running time.

A careful geometric proof can be quite cumbersome at times, so you are highly encouraged to use intuition and give illustrations whenever appropriate. However, beware of “proof by picture”: a poorly drawn figure might make you ignore special cases that are not apparent in your drawing. Try to keep your answers concise and to the point. Never give detailed code. Only give sufficient detail to convince us that your method and your running time analysis are complete and correct.

Unless stated otherwise, you can assume that objects are in general position, that is, no three points are collinear and no four points are cocircular. You may also assume that any geometric primitive that involves only a constant number of objects which are each of constant complexity can be computed in \(O(1)\) time.

Exercise 1 (G): [5 points] Let \(P\) be a set of points in the plane, and \(p \in P\) a given point. Give a linear-time algorithm that tests whether \(p\) is a vertex of the convex hull of \(P\).

Note: Only \(p\) and \(P\) are given as input, the convex hull of \(P\) is not given.

Exercise 2: [5 points] Let \(S\) be a set of \(n\) disjoint line segments in the plane, and let \(p\) be a point not on any of the line segments of \(S\). We wish to determine all line segments of \(S\) that \(p\) can see, that is, all line segments of \(S\) that contain some point \(q\) so that the open segment \(pq\) does not intersect any line segment of \(S\). The goal of this exercise is to develop an \(O(n \log n)\)-time algorithm for this problem.

Exercise 3 (G): [2+3 points] Let \(P\) be a set of \(n\) points in the plane. Let’s define a three-disk of \(P\) as a disk that contains at least three points of \(P\). The straightforward algorithm for computing the smallest three-disk takes \(O(n^3)\) time. The purpose of this exercise is to design a faster algorithm.

(a) For a point \(p = (p_x, p_y)\) in \(P\) let \(P'\) be the set of points with \(y\)-coordinate strictly larger than \(p_y\), i.e., the points above \(p\). Assume the smallest three-disk of \(P'\) has diameter \(\delta\). Now let \(R\) be the rectangle with lower left corner \((p_x - \delta, p_y)\) and upper right corner \((p_x + \delta, p_y + \delta)\). Prove that the number of points of \(P'\) that lies in \(R\) is bounded by a constant.

(b) Give an \(O(n \log n)\)-time algorithm that reports the smallest three-disk of \(P\). You can assume general position, in particular no two points have the same \(x\)- or \(y\)-coordinate.